A logo with blue text

AI-generated content may be incorrect.

Design, Development, and Deployment of a Secure Cloud-Based Meal and Diet Generator Using Angular and Spring Boot.

Module:- B9IS109 - WEB DEVELOPMENT FOR INFORMATION SYSTEMS

Group Members:

Kayis Rahman Badharakkanakath – 20035121

Kanimozhi Chandrasekar – 20049661

Barath Mani – 20058340

Aakash Dhakal – 20043599

Table of Contents

[1. Research and Planning 3](#_Toc196145950)

[Objective: 3](#_Toc196145951)

[1. Market Analysis 3](#_Toc196145952)

[Researched popular meal/diet planning apps like MyFitnessPal, Yazio, and Eat This Much. 3](#_Toc196145953)

[Identified similarities: user profiles, entry of dietary objectives, creation of meal plans, analysis of nutrients. 3](#_Toc196145954)

[Uncovered gaps in personalisation, onboarding simplicity, and security within some open-source applications. 3](#_Toc196145955)

[2. Technical Feasibility 3](#_Toc196145956)

[3. Deployment & Collaboration 3](#_Toc196145957)

[4. Choice of Framework and Technologies 4](#_Toc196145958)

[Frontend: Angular 4](#_Toc196145959)

[Why Angular? 4](#_Toc196145960)

[Key Libraries Used: 4](#_Toc196145961)

[Backend: Spring Boot 4](#_Toc196145962)

[Why Spring Boot? 4](#_Toc196145963)

[Key Tools Used: 4](#_Toc196145964)

[Database: PostgreSQL 5](#_Toc196145965)

[Deployment: Microsoft Azure 5](#_Toc196145966)

[Development Tools & Environments: 5](#_Toc196145967)

[5. UX Goals: 5](#_Toc196145968)

[6. Wireframing & Prototyping: 5](#_Toc196145969)

[2. Sequence Diagram 6](#_Toc196145970)

[Layout Flow: 6](#_Toc196145971)

[Integration Flow: 7](#_Toc196145972)

[Responsive Design: 7](#_Toc196145973)

[User Input Considerations: 7](#_Toc196145974)

[UI Styling Principles: 8](#_Toc196145975)

[3. Security Threats and Measures 8](#_Toc196145976)

[Common Security Threats Identified: 8](#_Toc196145977)

[4. Implemented Security Measures: 8](#_Toc196145978)

[Spring Security Integration 8](#_Toc196145979)

[Validation & Sanitisation 8](#_Toc196145980)

[CSRF Protection 9](#_Toc196145981)

[HTTPS Recommendation 9](#_Toc196145982)

[Authentication & Session Management 9](#_Toc196145983)

[5. Web Service Integration 9](#_Toc196145984)

[RESTful API Design with Spring Boot 9](#_Toc196145985)

[API Testing and Verification 9](#_Toc196145986)

[Angular–Spring Boot Integration Flow: 9](#_Toc196145987)

[6. Features of the Web App 9](#_Toc196145988)

[User Registration & Login (by Kayis) 9](#_Toc196145989)

[Meal Preference Form Submission (by Kani) 10](#_Toc196145990)

[Diet Generator (by Barath) 10](#_Toc196145991)

[Cloud Deployment (by Aakash) 10](#_Toc196145992)

[Cross-Browser & Mobile Support 10](#_Toc196145993)

[User Experience Additions 10](#_Toc196145994)

[8. Project / Presentation Resources 10](#_Toc196145995)

[7. Conclusion 10](#_Toc196145996)

[8. Project Outcomes 11](#_Toc196145997)

[9. Future Enhancements 11](#_Toc196145998)

[10. References 11](#_Toc196145999)

# 1. Research and Planning

## Objective:

This project aimed to develop a user-friendly, cloud-deployed web application named NutriMind. It helps users generate personalised diet plans based on their meal timing, health goals, and diet preferences. We aimed to establish a solid technical groundwork, a decent user experience, and safe practices.

**Initial Research:**

We conducted research under three primary domains:

## 1. Market Analysis

## Researched popular meal/diet planning apps like MyFitnessPal, Yazio, and Eat This Much.

## Identified similarities: user profiles, entry of dietary objectives, creation of meal plans, analysis of nutrients.

## Uncovered gaps in personalisation, onboarding simplicity, and security within some open-source applications.

## 2. Technical Feasibility

Compared full-stack frameworks:

* Angular + Spring Boot vs MERN.
* Choose Angular + Spring Boot for: Strong typing (TypeScript & Java)
* Rich form handling (Angular)
* Robust security & scalability (Spring Boot)

Evaluated databases and selected **PostgreSQL** for ACID compliance and schema flexibility.

## 3. Deployment & Collaboration

Decided on **Azure** for cloud deployment, based on student credits, documentation, and VM support.

Used **GitLab** for version control and collaborative development.

Choose **Intellij IDEA** as the primary IDE.

**Task Allocation & Planning:**

To ensure efficiency and accountability, our team of four divided responsibilities based on strengths and interests:

|  |  |
| --- | --- |
| **Member** | **Role/Task** |
| Kayis | Login & Registration Module with JWT token and Spring Security Configuration |
| Barath | Diet Generator Page and Chat GPT API integration. |
| Kani | Meal Preference Form Page with multiple accordion |
| Aakash | API and behavioural Testing, Azure Deployment & Hosting with SSL and https |

Each member was responsible for both the frontend and backend components of their module, including API integration and testing.

## 4. Choice of Framework and Technologies

### Frontend: Angular

### Why Angular?

Angular provides a **modular architecture**, **two-way data binding**, and **dependency injection**, making it suitable for scalable enterprise-level apps.

**Built-in form handling and validation**, ideal for login, registration, and meal form input.

TypeScript support enables **better debugging and maintainability**.

### Key Libraries Used:

* @angular/forms – for form validation.
* ngx-toastr – for user-friendly notifications.
* Angular Material – for responsive UI components.

### Backend: Spring Boot

### Why Spring Boot?

Lightweight and fast backend framework using Java.

Comes with embedded servers (Tomcat), reducing deployment hassles.

Built-in **security, REST API support**, and **integration with Postgresql**.

Enables **MVC-based clean separation** of concerns.

### Key Tools Used:

* Spring Web, Spring Data JPA, Spring Security.
* Lombok – to reduce boilerplate code.
* ModelMapper – for DTO and Entity mapping.

### Database: PostgreSQL

Open-source relational database that supports **complex queries**, **JSON data types**, and strong **transactional guarantees**.

Useful for storing structured data like user info, dietary preferences, and meal records.

### Deployment: Microsoft Azure

Azure VM chosen for its **free student credits**, **scalability**, and **easy SSH access**.

Deployed backend on Azure Virtual Machine (Ubuntu Server).

The PostgreSQL database is also hosted on Azure.

### Development Tools & Environments:

|  |  |
| --- | --- |
| **Tool** | **Purpose** |
| IntelliJ IDEA | Frontend development (Angular)  Backend development (Spring Boot) |
| Postman | API testing |
| Git & GitLab | Version control and collaboration |
| Azure CLI | VM management and deployment |

**3. UX Design**

**Target Audience:**

* Health-conscious individuals.
* People seeking quick, easy-to-generate meal plans.
* Users with specific dietary restrictions (e.g., vegetarian, keto, gluten-free).

## 5. UX Goals:

* Minimise cognitive load: Simple input forms and clearly labelled steps.
* Fast navigation: One-click access to main features like generating a diet or adding meals.
* Accessibility: High contrast UI, legible fonts, and keyboard-friendly controls.

## 6. Wireframing & Prototyping:

Designed basic wireframes using **Draw.io** to visualise layout and screen flow.

Emphasis on minimalistic design and intuitive navigation.

# 2. Sequence Diagram

A diagram of a diet plan

AI-generated content may be incorrect.

## Layout Flow:

A diagram of a spring boot block

AI-generated content may be incorrect.

## Integration Flow:

A diagram of a computer system

AI-generated content may be incorrect.

## Responsive Design:

Used **Angular Flex-Layout** and **CSS Grid/Flexbox** to ensure compatibility across:

* Desktop (main target)
* Tablets
* Mobile devices

## User Input Considerations:

* Prevent invalid form submissions with real-time validation feedback.
* Provide placeholder examples and tooltip help for dietary inputs.

## UI Styling Principles:

* **Color Scheme**: Light green + white tones to promote freshness and health.
* **Font Choice**: Sans-serif for readability.
* **Icons**: Used recognizable food and health icons for quick understanding.
* **Notable Features for UX:**
* **Success toast** after plan generation for positive reinforcement.
* **Error alerts** when diet generation fails due to missing input.
* **Loading spinners** during diet creation to enhance perceived speed.

# 3. Security Threats and Measures

## Common Security Threats Identified:

|  |  |  |
| --- | --- | --- |
| **Threat** | **Description** | **Impact** |
| **SQL Injection** | Malicious input via form fields to manipulate database queries. | Data theft, unauthorized access. |
| **Cross-Site Scripting (XSS)** | Injection of malicious scripts in form fields. | Data leakage, session hijacking. |
| **Cross-Site Request Forgery (CSRF)** | Exploiting an authenticated user’s session to perform unintended actions. | Account compromise, unauthorized data manipulation. |
| **Brute Force Attacks** | Repeated attempts to guess login credentials. | Unauthorized access. |
| **Insecure APIs** | Exposure of sensitive endpoints or lack of proper authentication. | Data breaches, loss of control. |

# 4. Implemented Security Measures:

## Spring Security Integration

* Enforced role-based access control (ROLE\_USER, ROLE\_ADMIN) for backend routes.
* Used **BCrypt hashing** for password encryption before storing in PostgreSQL.
* Disable unused or vulnerable HTTP methods (PUT, DELETE on auth endpoints).

## Validation & Sanitisation

* All user input fields (diet form, login, registration) are:
* Validated on the **frontend using Angular Reactive Forms**.
* Re-validated and sanitised on the **backend using annotations** like @Valid and input checks.

## CSRF Protection

* Enabled CSRF tokens via Spring Security for forms and API endpoints.
* Tokens passed in headers during Angular–Spring communication.

## HTTPS Recommendation

* We did set up HTTPS using a valid SSL certificate on the Azure VM.
* Ensures end-to-end encryption between the user and server.

## Authentication & Session Management

* Implemented **JWT (JSON Web Tokens)** for secure, stateless authentication.
* Tokens are stored securely in **HTTP-only cookies** or Angular memory (for this project, stored in localStorage with are expiration timeout for demo purposes).

# 5. Web Service Integration

## RESTful API Design with Spring Boot

Designed and exposed a REST API to allow secure communication between the Angular frontend and Spring Boot backend.

Followed REST conventions:

* GET /meal-pref – fetch existing meal preference.
* POST /meal-pref – add a new meal preference.
* POST /diet/generate – initiate diet plan generation.
* POST /auth/register – user registration.
* POST /auth/login – user authentication.

## API Testing and Verification

* All endpoints were tested using **Postman** before connecting to Angular.
* Added appropriate **HTTP status codes**:
* 200 OK, 201 Created, 400 Bad Request, 401 Unauthorised, etc.

## Angular–Spring Boot Integration Flow:

* Angular used HttpClientModule with interceptors for:
* Adding JWT tokens to request headers.
* Handling global errors and session expiration.
* Secure API Access
* Protected backend APIS with Spring Security filters.
* Added custom exceptions and response handlers to ensure clear error messages without exposing stack traces.

# 6. Features of the Web App

## User Registration & Login (by Kayis)

* Secure registration form with validations (email, password strength).
* Login mechanism using JWT-based authentication.
* Role assignment upon registration (default USER).

## Meal Preference Form Submission (by Kani)

* A responsive and validated form where users can enter meal details (e.g., food name, quantity, time).
* Forms support CRUD operations: users can **add**, **view**, and **delete** meals.
* Data is stored securely in a **PostgreSQL** database.

## Diet Generator (by Barath)

* Users’ input preferences such as health goals (weight gain/loss), diet type (vegetarian, high-protein, etc.).
* The system returns a **customised diet plan** generated from a predefined list of food categories using algorithmic logic.
* Output is visualised in a card layout with colour-coded nutrients.

## Cloud Deployment (by Aakash)

* The entire project was deployed to a **Microsoft Azure Virtual Machine**.
* **PostgreSQL** database hosted on an Azure VM.
* Backend: Hosted Spring Boot application with reverse proxy via **Nginx**.
* Frontend: Angular build deployed and served using **Nginx** static hosting.

## Cross-Browser & Mobile Support

* Responsive design using Angular’s grid layout and Flexbox.
* Compatible with Chrome, Firefox, Edge, and mobile browsers.

## User Experience Additions

* Toast notifications for success/error.
* Loading indicators while generating diets.
* Smooth routing with Angular Router and lazy loading.

# 8. Project / Presentation Resources

* [GitHub link: https://github.com/kayis-rahman/nutriment](https://github.com/kayis-rahman/nutriment)
* [Presentation slide: https://github.com/kayis-rahman/nutriment/blob/main/Nutrimind\_Presentation.pptx](https://github.com/kayis-rahman/nutriment/blob/main/Nutrimind_Presentation.pptx)
* [Individual contribution: https://github.com/kayis-rahman/nutriment/blob/main/Individual\_contributions.docx](https://github.com/kayis-rahman/nutriment/blob/main/Individual_contributions.docx):
* [Video Recording: https://mydbs-my.sharepoint.com/:v:/g/personal/20035121\_mydbs\_ie/EeMmalr4pxpJoQINJl3hWPYBn95y1eXV9Y1ngpSkemoPxQ?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=I7RgjN](https://mydbs-my.sharepoint.com/:v:/g/personal/20035121_mydbs_ie/EeMmalr4pxpJoQINJl3hWPYBn95y1eXV9Y1ngpSkemoPxQ?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=I7RgjN)

# 7. Conclusion

We aimed to develop an end-to-end working, cloud-based meal and diet planning web application with as much emphasis as possible on user experience, security, and scalability. While coding, we strictly adhered to contemporary software development methodologies, including:

* Modular design using Angular for the frontend and Spring Boot for the backend.
* Secure authentication and form handling.
* RESTful API integration with proper testing and validation.
* Responsive, user-friendly UI for both desktop and mobile users.
* Deployment of a production-ready system on **Microsoft Azure**.

# 8. Project Outcomes

* Delivered a reliable and secure multi-role application.
* Successfully collaborated as a team, maintaining clear ownership of features.
* Applied web security principles and cloud computing concepts in a practical setting.
* Gained hands-on experience in CI/CD concepts, cloud hosting, and API security.

# 9. Future Enhancements

* Add health tracking using fitness APIS or wearable integration.
* Enable user profile customisation with calorie tracking history.
* Use AI/ML for dynamic diet suggestions based on user trends.
* Integration with nutrition APIS for up-to-date meal suggestions.

# 10. References

Angular.io (2024) *Introduction to Angular*. [online] Available at: <https://angular.io/guide/what-is-angular> [Accessed 17 Apr. 2025].

Baeldung (2024) *Guide to Spring Boot*. [online] Available at: <https://www.baeldung.com/spring-boot-start> [Accessed 17 Apr. 2025].

BCrypt (2023) *BCrypt Password Hashing*. [online] Available at: <https://github.com/jeremyh/jBCrypt> [Accessed 17 Apr. 2025].

Draw.io (2024) *Flowchart Maker & Online Diagram Software*. [online] Available at: <https://app.diagrams.net/> [Accessed 17 Apr. 2025].

Microsoft (2024) *Azure Virtual Machines Documentation*. [online] Available at: <https://learn.microsoft.com/en-us/azure/virtual-machines/> [Accessed 17 Apr. 2025].

OWASP (2023) *OWASP Top 10 Security Threats*. [online] Available at: <https://owasp.org/www-project-top-ten/> [Accessed 17 Apr. 2025].

PostgreSQL.org (2024) *The World’s Most Advanced Open Source Relational Database*. [online] Available at: <https://www.postgresql.org/> [Accessed 17 Apr. 2025].

Spring.io (2024) *Spring Boot Reference Documentation*. [online] Available at: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle>/ [Accessed 17 Apr. 2025].

Toastr (2024) *ngx-toastr – Angular Toast Notifications*. [online] Available at: <https://www.npmjs.com/package/ngx-toastr> [Accessed 17 Apr. 2025].

W3Schools (2024) *CSS Flexbox and Grid Layouts*. [online] Available at: <https://www.w3schools.com/css/css3_flexbox.asp> [Accessed 17 Apr. 2025].

Postman (2024) *API Platform for Building and Using APIs*. [online] Available at: <https://www.postman.com/> [Accessed 17 Apr. 2025].

JWT.io (2024) *Introduction to JSON Web Tokens*. [online] Available at: <https://jwt.io/introduction/> [Accessed 17 Apr. 2025].

GitLab (2024) *Version Control with GitLab*. [online] Available at: <https://about.gitlab.com/> [Accessed 17 Apr. 2025].

Material.angular.io (2024) *Angular Material UI Components*. [online] Available at: <https://material.angular.io/> [Accessed 17 Apr. 2025].

IntelliJ IDEA (2024) *JetBrains IntelliJ IDEA Overview*. [online] Available at: <https://www.jetbrains.com/idea/> [Accessed 17 Apr. 2025].

ModelMapper (2024) *Simplify Object Mapping in Java*. [online] Available at: <http://modelmapper.org/> [Accessed 17 Apr. 2025].

Lombok (2024) *Project Lombok Features*. [online] Available at: <https://projectlombok.org/> [Accessed 17 Apr. 2025].

Eat This Much (2024) *Automated Diet Planner*. [online] Available at: <https://www.eatthismuch.com/> [Accessed 17 Apr. 2025].

MyFitnessPal (2024) *Calorie Counter and Food Diary*. [online] Available at: <https://www.myfitnesspal.com/> [Accessed 17 Apr. 2025].

Yazio (2024) *Nutrition Tracker and Calorie Counter App*. [online] Available at: <https://www.yazio.com/en> [Accessed 17 Apr. 2025].